SASS

Construire dynamiquement vos styles CSS

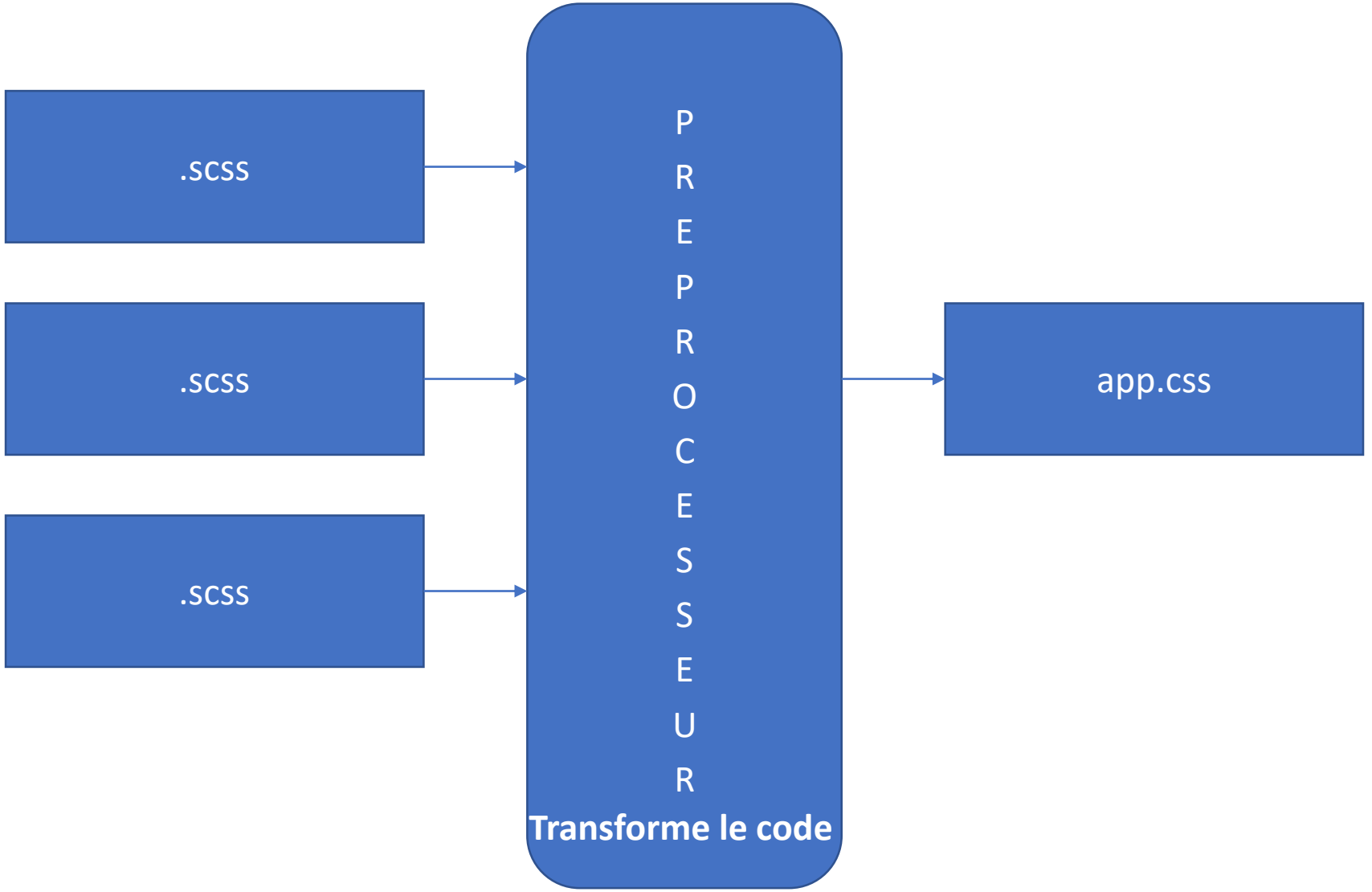
OBJECTIFS DE LA FORMATION

Créer des feuilles de style CSS à l'aide de SASS

A l'issue de la formation vous serez en mesure de :

- ✓ Installer SASS
- ✓ Utiliser les syntaxes de bases
- ✓ Ecrire des règles de style en SCSS
- ✓ Créer et utiliser les mixins
- ✓ Créer et utiliser les lists et maps
- ✓ Créer et utiliser les fonctions
- ✓ Factoriser le code
- ✓ Créer, importer et paramétrer un module

SASS



Ajoute des fonctionnalités

- ✓ Ecriture plus rapide du CSS
- ✓ Meilleure organisation du code
- ✓ Eviter la répétition
- ✓ Variables
- ✓ Boucles
- ✓ Structures conditionnelles
- ✓ Mixins
- ✓ Fonctions
- ✓ imbrication
- ✓ Héritage…

Documentation officielle: https://sass-lang.com/

DIFFERENCE ENTRE LESS ET SASS

	LESS	SASS
Variables	`@noir: black;`	`$noir: black`

Mixins	`.encadrer() {`
	`  border: 1px solid @noir;`
	`}`
		`@mixin encadrer()`
		`  border: 1px solid $noir`

Mixins avec paramètre(s)	`.encadrer(@largeur-bordure: 2px) {`
	`  border: @largeur-bordure solid @noir;`
	`}`
		`@mixin encadrer($largeur-bordure: 2px)`
		`  border: $largeur-bordure solid $noir`

Fonctions	`lighthen(#ff0000, 10%);` `darken(#ff0000, 20%);`	`lighthen(#ff0000, 10%);` `darken(#ff0000, 20%);`

Less : Leaner Style Sheets
Sass : Syntactically Awesome Style Sheets

DIFFERENCE ENTRE SASS ET SCSS

SASS	SCSS
$noir: black	$noir: black;

@mixin encadrer($largeur-bordure: 2px)
  border: $largeur-bordure solid $noir
	@mixin encadrer($largeur-bordure: 2px) {
	  border: $largeur-bordure solid $noir;
	}

Sass est un préprocesseur.
Il peut utiliser deux langages, le Sass et le SCSS.
Sass utilise l'indentation et les sauts de ligne pour délimiter les blocs qui constituent le code.
Sass ne comporte pas d'accolades et points-virgules contrairement à SCSS.
Dans cette formation, nous utiliserons le langage SCSS.

INSTALLER SASS

Plusieurs possibilités pour installer Sass

En tant que module node (pré-requis node.js) :

```
$ npm install –g sass
```

Fichiers source du projet

Sur le site (https://sass-lang.com/install) au paragraphe install anywhere (Standalone)

Avec le gestionnaire de paquet

```
$ apt |yum| brew | choco | whatever install sass
```

COMPILER UN FICHIER

Pour compiler un fichier, une seule fois

```
$ sass input.scss output.css
```

Pour re-compilé un fichier automatiquement

```
$ sass --watch input.scss output.css
```

Pour re-compilé un répertoire automatiquement:

```
$ sass --watch css:css
$ sass --watch app/sass:public/css
```

Pour compiler sans source-map et minifiée

```
$ sass --no-source-map --style=compressed scss/input.scss css/output.css
```

TIPS

Pour plus de confort et accroître votre productivité, vous pouvez utiliser l'extension :

- Live Server : permet de recharger automatiquement votre navigateur

Pour créer plus rapidement vos fichiers HTML, vous pouvez utiliser Emmet

Documentation : https://docs.emmet.io/abbreviations/syntax/

EXTENSIONS LIVE SASS COMPILER

Vous pouvez également compiler votre code directement sur Visual Studio Code avec Live Sass Compiler.

L'extension Live Sass Compiler de Ritwick Dey n'est plus maintenue depuis mi-2018 et pourrait ne pas reconnaître certaines fonctionnalités de Sass actuelles.
Une toute nouvelle extension créée par Glenn Marks, basée sur le travail de Ritwick Dey est disponible sur VSCode.
Prévilégiez cette dernière.

<u>Paramétrage de l'extension</u>
Fichier->Préférences->Paramètres
Fenêtre paramètre -> onglet espace de travail : extensions -> live sass compiler -> Live Sass Compile › Settings: Formats-> modifier dans settings.json

```
{
    "liveSassCompile.settings.formats": [
    {
            "format": "expanded", // Autre valeur possible compressed
            "extensionName": ".css", //Autre valeur possible .min.css
            "savePath": "public/css", //répertoire de destination
        }
    ]
}
```

ATELIER

Installer Sass et compiler un premier fichier

UTILISER LES SYNTAXES DE BASE

DECLARATION DE VARIABLES

Différence entre SASS et SCSS

SASS	SCSS

$font-stack : Helvetica, sans-serif
$primary-color: #333
 	
body
  font: $font-stack
  color: $primary-color
	$font-stack: Helvetica, sans-serif;
	$primary-color: #333;
	
	body {
	  font: $font-stack;
	  color: $primary-color;
	}

LES TYPES DE DONNEES

Types de données	Exemples
Nombre : 2 composants (nombre + [unités])	Nombre avec unité : 30px, 1em Nombre sans unité : 30
Chaîne de caractères	"hello", ’bonjour’, une chaîne avec espace
booléen	true, false
null	null
liste	$list1: valeurs, séparées, avec, virgules
mapping	$map1: (clé1: val1, clé2: val2)

Colors	#CDCDCD
	rgb(0,0,0)
	rgba(0,0,0,0.5)

LES COMMENTAIRES

Les commentaires du fichier SCSS seront inclus ou non dans le fichier CSS selon la syntaxe utilisée.

```
//Ce commentaire ne sera pas inclus dans le fichier CSS

/* Mais celui-ci sera inclus sauf dans le mode compressé */

/*!  Ce commentaire sera inclus même dans le mode compressé */
```

Les opérateurs arithmétiques

Opérateur	Opération
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

Les opérateurs de comparaison

Opérateur	Signification
==	Egalité
!=	Différence
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égale
>=	Supérieur ou égale

LES OPERATEURS

Les opérateurs de chaines de caractères

Opérateur	Résultat
$chaine1: "ch1"; $chaine2: "ch2"; $chaine3: "ch1"+"ch2"; @debug $chaine3;	$ch1ch2
@debug "ch1"+"ch2";	$ch1ch2

Les opérateurs logiques

Opérateur	Résultat
@debug not true;	False
@debug not false;	true
@debug true and true;	True
@debug true and false;	False
@debug true or false;	True
@debug false or false;	false

LA PORTEE DES VARIABLES 1/2

Les variables déclarées au niveau supérieur d'une feuille de style sont globales.
Celles qui sont déclarées dans des blocs (accolades dans SCSS ou code indenté dans Sass) ne sont accessibles que dans le bloc où elles ont été déclarées.

Exemple	Résultat

`$global-variable: 16px;`
 	
`p{`
`  font-size: $global-variable;`
`  $blue-color: blue;`
`  color: $blue-color;`
`}`
 	
`a{`
`  font-size: $global-variable;`
`  color: $blue-color;`
`//Variable $blue-color inconnue accessible que dans le`
`//bloc p`
`}`
	Erreur de compilation, la variable $blue-color n'existe que dans le bloc p

LA PORTEE DES VARIABLES 2/2

Si vous devez définir la valeur d'une variable globale dans une portée locale, vous pouvez utiliser le drapeau !global.

Exemple	Résultat

`$global-variable: 16px;`
 	
`p{`
`  font-size: $global-variable;`
`  $blue-color: blue !global;`
`  color: $blue-color;`
`}`
 	
`a{`
`  font-size: $global-variable;`
`  color: $blue-color;`
`}`
	Avec le drapeau !global, la variable $blue-color est bien accessible dans le bloc a

LES STRUCTURES CONDITIONNELLES

SCSS

```scss
$type: monster;

p {
    @if $type==ocean {
        color: blue;
    } @else if $type==matador {
        color: red;
    } @else if $type==monster {
        color: green;
    } @else {
        color: black;
    }
}

//Opérateur ternaire
body{
    $val: 0;
    width: if($val == 0, 768px, 1200px);
}
```

CSS

```css
p {
  color: green;
}

/*Si $val est égal à 0, width: 768px sinon
width: 1200px */

body {
    width: 768px;
}
```

LES BOUCLES FOR

SCSS

```scss
$columns: 6;

//through de 1 à 6
@for $i from 1 through $columns {
    .columns-#{$i} {
        width: (100% / $i);
    }
}

//to de 1 à 5
@for $i from 1 to $columns {
    .columns-#{$i} {
        width: (100% / $i);
    }
}
```

CSS

```css
.columns-1 {
  width: 100%;
}
.columns-2 {
  width: 50%;
}
.columns-3 {
  width: 33.3333333333%;
}
.columns-4 {
  width: 25%;
}
.columns-5 {
  width: 20%;
}
.columns-6 {
  width: 16.6666666667%;
}
```

BOUCLE EACH

SCSS

```scss
@each $i in html-css, javascript, sass, jquery {
    .fond-formation-#{$i} {
        background-color: blue;
    }
}
```

CSS

```css
.fond-formation-html-css {
  background-color: blue;
}

.fond-formation-javascript {
  background-color: blue;
}

.fond-formation-sass {
  background-color: blue;
}

.fond-formation-jquery {
  background-color: blue;
}
```

BOUCLE EACH

Affectation multiple

SCSS

```scss
@each $service, $color in
                      (html-css, blue),
                      (javascript, green),
                      (sass,pink),
                      (jquery, aqua) {
    .fond-formation-#{$service} {
        background-color: $color;
    }
}
```

CSS

```css
.fond-formation-html-css {
  background-color: blue;
}

.fond-formation-javascript {
  background-color: green;
}

.fond-formation-sass {
  background-color: pink;
}

.fond-formation-jquery {
  background-color: aqua;
}
```

LES BOUCLES WHILE

SCSS

```scss
$i: 1;

@while $i<=6 {
   .columns-#{$i} {
      width: (100% / $i);
   }
   $i: $i + 1;
}
```

CSS

```css
.columns-1 {
  width: 100%;
}
.columns-2 {
  width: 50%;
}
.columns-3 {
  width: 33.3333333333%;
}
.columns-4 {
  width: 25%;
}
.columns-5 {
  width: 20%;
}
.columns-6 {
  width: 16.6666666667%;
}
```

ATELIER

Utiliser des variables pour définir les couleurs d’un thème

ECRIRE DES RÈGLES DE STYLE

DRY

Don't repeat yourself

Ne vous répétez pas (don't repeat yourself en anglais, aussi désigné par l'acronyme DRY) est une philosophie en programmation informatique consistant à éviter la redondance de code au sein d'une application afin de faciliter la maintenance, le test, le débogage et les évolutions.

Source : https://fr.wikipedia.org/wiki/Ne_vous_r%C3%A9p%C3%A9tez_pas

L' IMBRICATION DES RÈGLES CSS

SCSS

```scss
$dark-color: #222;
$light-color: #ddd;
$brand-color: #5a5;
table {
    color: $dark-color;
    // une règle imbriquée applique la relation d'ascendance
    th {
        background-color: $dark-color;
        color: $light-color;
    }
    td {
        color: lighten($dark-color, 20);
        a {
            display: block;
            background-color: $brand-color;
            &:hover {
                background-color: lighten($brand-color, 20);
            }
        }
    }
}
```

CSS

```css
table {
  color: #222;
}
table th {
  background-color: #222;
  color: #ddd;
}
table td {
  color: #555555;
}
table td a {
  display: block;
  background-color: #5a5;
}
table td a:hover {
  background-color: #99cc99;
}
```

LE SELECTEUR PARENT « & »

SCSS	CSS	Commentaires

a{
:hover{
color: blue;
}
}
	a :hover {
	color: blue;
	}
		Le code est compilé mais ne s'exécute pas :
		
		a :hover (espace entre la balise et la pseudo-classe)

a {
&:hover {
color: blue;
}
}
	a:hover {
	color: blue;
	}
		Avec le &, plus d'espace, le code est correct

.button {
body.page-about & {
color: black;
}
}
	body.page-about .button {
	color: black;
	}
		Permet de placer le sélecteur parent ou vous le désirez.

LES COMBINATEURS

Les sélecteurs combinateurs combinent différents sélecteurs de façon à leur donner une relation utile et l'emplacement du contenu dans le document.

Type de combinateur	Exemple	Définition
Combinateur descendant	body article p	Le combinateur descendant— en général représenté par un seul espace (" "), combine deux sélecteurs de sorte que les éléments choisis par le second sélecteur sont sélectionnés s'ils ont un élément ancêtre (parent, parent de parent, parent de parent de parent, etc...) qui correspond au premier sélecteur. Les sélecteurs qui utilisent un combinateur descendant sont appelés sélecteurs descendants.
Combinateur enfant	article > p	Le combinateur enfant (>) est placé entre deux sélecteurs CSS. Il correspond uniquement aux éléments correspondant au second sélecteur qui sont les enfants directs des éléments correspondants au premier. Les éléments descendants plus bas dans la hiérarchie ne correspondent pas.
Combinateur frère adjacents (adjacent sibling selector)	p + img	Le sélecteur de frère adjacent (+) est utilisé pour sélectionner quelque chose s'il est juste à côté d'un autre élément au même niveau de la hiérarchie.
Combinateurs général de frères (general sibling selector)	p ~ img	Si vous souhaitez sélectionner les frères d'un élément même s'ils ne sont pas directement adjacents, vous pouvez utiliser le combinateur général de frères (~).

CSS – LES COMBINATEURS - EXEMPLE

Soit le code html suivant :

```
<header>
        <h1>Titre 1 dans le header</h1>
        <h2>Titre 2 dans le header</h2>
    </header>
    <main>
        <h2>Titre 2 dans le main</h2>
        <p>Paragraphe 1</p>
        <p>Paragraphe 2</p>
        <p>Paragraphe 3</p>
    </main>
```

CSS – LES COMBINATEURS - EXEMPLE

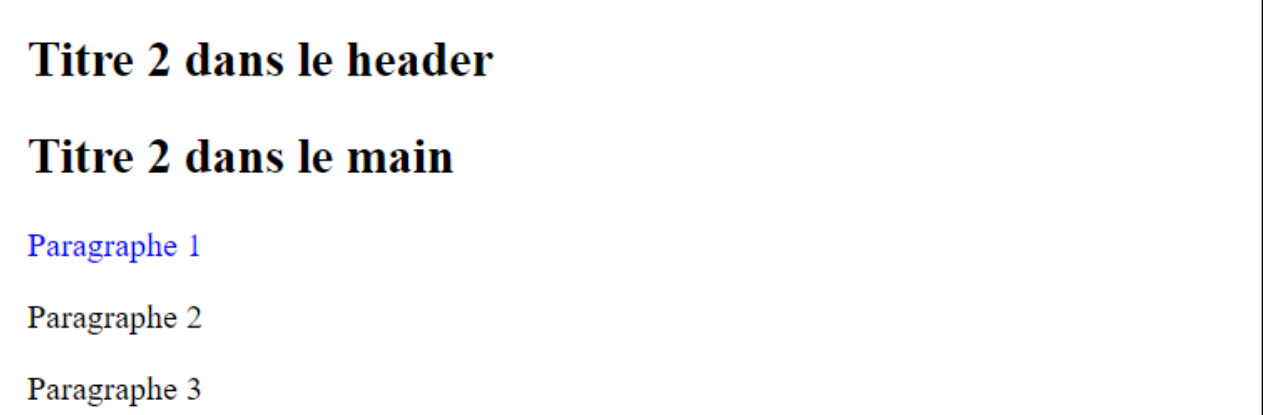
Combinateur descendant	`body h2 {`
	`  color: blue;`
	`}`
		Titre 1 dans le header
		Titre 2 dans le header
		Titre 2 dans le main
		Paragraphe 1
		Paragraphe 2
		Paragraphe 3

Combinateur enfant	`header > h2 {`
	`  color: blue;`
	`}`
		Titre 1 dans le header
		Titre 2 dans le header
		Titre 2 dans le main
		Paragraphe 1
		Paragraphe 2
		Paragraphe 3

CSS – LES COMBINATEURS - EXEMPLE

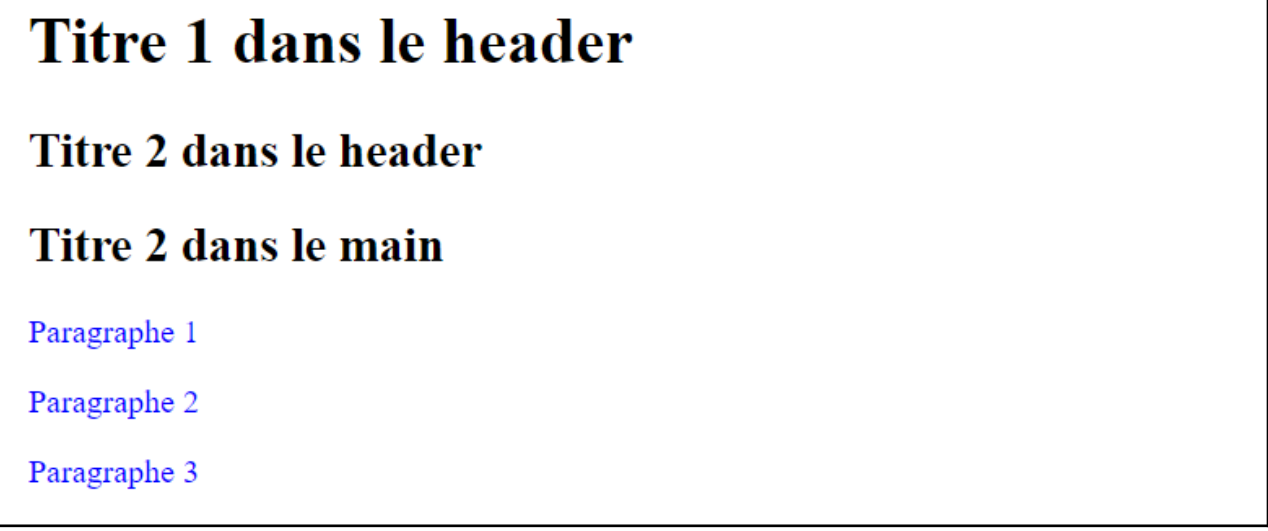
Combinateur frère adjacents
(adjacent sibling selector)

```
h2+p {
  color: blue;
}
```



Combinateurs général de frères
(general sibling selector)

```
h2~p {
   color: blue;
}
```



LES COMBINATEURS

SCSS

```scss
ul > {
    li {
        list-style-type: none;
    }
}

h2 + {
    p {
        border-top: 1px solid gray;
    }
}

p ~ span {
    opacity: 0.8;
}
```

CSS

```css
ul > li {
  list-style-type: none;
}


h2 + p {
  border-top: 1px solid gray;
}


p ~ span {
  opacity: 0.8;
}
```

UTILISER LES INTERPOLATIONS DE CHAINE

SCSS	CSS	Commentaires

`$name: 'chat';`
 	
`body{`
`  background-image:`
`url("/images/$name.jpg");`
`}`
	`body {`
	`  background-image:`
	`url("/images/$name.jpg");`
	`}`
		Le code est compilé mais la variable $name n'est pas le résultat que l'on attendait

`$name: 'chat';`
 	
`body{`
`  background-image:`
`url("/images/#{$name}.jpg");`
`}`
	`body {`
	`  background-image:`
	`url("/images/chat.jpg");`
	`}`
		Pour que $name soit remplacé par chat dans le code CSS, il faut avoir recours à l'interpolation.
		Pour cela, au lieu d'écrire $name,
		Il faut écrire **#{**$name**}**

IMBRICATION DE PROPRIETES

SCSS	Code CSS

p{
font: {
family: Georgia, serif;;
size: 1.2rem;
}
}
	p {
	font-family: Georgia, serif;
	font-size: 1.2rem;
	}

L'imbrication de propriétés se fait grâce aux : après font

ATELIER

Mettre en forme une barre de navigation

UTILISER LES MIXINS

DÉCLARER UNE MIXIN

Les mixins Sass permettent de créer des groupes de déclarations CSS pour une utilisation ultérieure.

Mixin sans paramètre

```
@mixin nomMixin() {
    //Votre code
}
```

SCSS

```
@mixin frame(){
    border:5px solid black;
    border-radius: 5px 5px 5px 5px;
    background-color: blue;
    width: 400px;
}
```

Inclusion d'une mixin

```
div {
   @include frame();
}
```

CSS

```
div {
  border: 5px solid black;
  border-radius: 5px 5px 5px 5px;
  background-color: blue;
  width: 400px;
}
```

To frame : encadrer

ARGUMENTS

Création d'une mixin (argument obligatoire)

```
@mixin button($color, $background-color,$hover-color,$hover-background-color,$font-size) {
    font-size: $font-size;
    color: $color;
    background-color: $background-color;
    &:hover {
        color: $hover-color;
        background-color: $hover-background-color;
    }
}
```

Obligation de déclarer tous les paramètres lors de l'inclusion sinon erreur de compilation

```
@mixin button($color, $background-color,$hover-color,$hover-background-color,$font-size)
button{
    @include button($color: #D59595, $background-color: #CCCCCC,$hover-color:
#CCCCCC,$hover-background-color: #D59595,$font-size: 30px);
}
```

INCLURE UNE MIXIN DANS UNE CLASSE

Considérant la mixin suivante :

```scss
@mixin arrondi() {
    border-radius: 20px;
}
```

Inclusion de la mixin dans une classe

SCSS	CSS
`.btn { @include arrondi(); }`	`.btn { border-radius: 20px; }`

ARGUMENTS

Création d'une mixin (argument optionnel)

Si vous vous voulez déclarer des arguments optionnels, il faut les déclarer par défaut.

```
@mixin button($color: #D59595, $background-color: #CCCCCC,$hover-color: #CCCCCC,$hover-background-color: #D59595,$font-size: 30px) {
    font-size: $font-size;
    color: $color;
    background-color: $background-color;
    &:hover {
        color: $hover-color;
        background-color: $hover-background-color;
    }
}
```

Sass "remplit" les arguments de gauche à droite : on ne peut pas renseigner une valeur pour $hover-color, sans donner au préalable une valeur à $color. Il est important de faire attention à l'ordre.

Arguments nommées

Si tous les paramètres sont déclarés par défaut, vous pouvez utiliser les paramètres nommés pour modifier spécifiquement un ou plusieurs paramètres.

Pour cela, vous devez nommer les noms de variables à l'inclusion de la mixin.

L'ordre et le nombre de paramètres nommés n'ont aucune importance.

```
button{
    @include button($color: black);
}
```

```
button{
    @include button($font-size: 60px, $hover-color: red);
}
```

UTILISER LE BLOC DE CONTENU

SCSS	CSS

`@mixin hover {`
`  &:hover {`
`  @content;`
`  }`
`}`	

`.button {`
`  border: 1px solid black;`
`  @include hover {`
`  border-width: 2px;`
`  }`
`}`
	`.button {`
	`  border: 1px solid black;`
	`}`
	`.button:hover {`
	`  border-width: 2px;`
	`}`

ATELIER

Utiliser les mixins pour créer une grille responsive

LISTS ET MAPS

DIFFERENCE ENTRE LIST ET MAP

LIST	MAP

Plusieurs syntaxes possibles :
 	
`$syntax-01: 1rem 2rem 3rem 4rem;`
`$syntax-02: 1rem, 2rem, 3rem, 4rem;`
`$syntax-03: (1rem 2rem 3rem 4rem);`
`$syntax-04: (1rem, 2rem, 3rem, 4rem);`
	`$font-size: (logo:7rem, heading:5rem, project-heading:4rem, label:2rem);`
	
	`//Ecriture plus lisible`
	`$font-size: (`
	`  logo:7rem,`
	`  heading:5rem,`
	`  project-heading:4rem,`
	`  label:2rem`
	`);`

`@debug $syntax-01; // 1rem, 2rem, 3rem, 4rem` `@debug list.nth($syntax-01, 2); //2rem`	`@debug map-get($font-size, label); // 2rem`

PARCOURIR LES VALEURS A L'AIDE DE @EACH

Map

```
$icons: (
  "eye": "\f112",
  "start": "\f12e",
  "stop": "\f12f",
);

@each $name, $glyph in $icons {
  .icon-#{$name}:before {
    display: inline-block;
    font-family: "Icon Font";
    content: $glyph;
  }
}
```

List

```
@use "sass:list";

$listeA: ("eye": "\f112"),
  ("start": "\f12e"),
  ("stop": "\f12f");

@each $name, $glyph in $listeA {
  .icon-#{$name}:before {
    display: inline-block;
    font-family: "Icon Font";
    content: $glyph;
  }
}
```

Pour utiliser le module sass:list, il faut décaler la directive

`@use "sass:list" en début de fichier`

Action	Instruction
Accéder à un élément	`@debug list.nth($sizes, 2 ); // 50px`
Ajouter une valeur	`$sizes: append($sizes, 100px);`
Trouver un élément	`@debug list.index($sizes, 40px); //1`

MANIPULER LES DONNEES AVEC SAAS:MAP

- Pour utiliser le module sass:map, il faut décaler la directive

`@use "sass:map"` en début de fichier.

$font-size: (logo:7rem, heading:5rem, project-heading:4rem, label:2rem);

Extraire une valeur	`@debug map-get($font-size, label); // 2rem`
Ajouter une valeur	`$font-size: map.set($font-size,"body", 16px);`
Modifier une valeur	`$font-size: map.set($font-size,logo, 60px);`
Supprimer une valeur	`$font-size: map.remove($font-size,logo);`
Fusionner deux maps	`$map3: map.merge($font-size, $map2);`

TRANSMETTRE UNE LISTE DE VALEURS A UNE MIXIN

```scss
$list-sizes: 40px, 50px;

@mixin icon-size($sizes: $list-sizes) {
    @each $size in $sizes {
        .icon-#{$size} {
            font-size: $size;
            height: $size;
            width: $size;
        }
    }
}
```

```scss
.icon-phone{
    @include icon-size;
}
```

```scss
.icon-phone .icon-40px {
  font-size: 40px;
  height: 40px;
  width: 40px;
}
.icon-phone .icon-50px {
  font-size: 50px;
  height: 50px;
  width: 50px;
}
```

ATELIER

Utiliser les listes et les maps pour décliner un composant CSS dans plusieurs couleurs ou plusieurs tailles

UTILISER LES FONCTIONS

DÉCLARER ET UTILISER UNE FONCTION

SCSS

```scss
@function nomFonction($param1, $param2,..) {
  //Votre code
    @return $valeur;
  }
//$paramètre... permet d'appeler la fonction
avec un nombre indéterminé de paramètres
@function sum($numbers...) {
    $sum: 0;
    @each $number in $numbers {
        $sum: $sum + $number;
    }
    @return $sum;
}

.micro {
    width: sum(50px, 30px, 100px);
}
```

CSS

```css
.micro {
  width: 180px;
}
```

FONCTIONS PREDEFINIES

Quelques fonctions prédéfinies de Sass:

- darken()
- lighten()
- random()
- round()…

RETOURNER DES VALEURS

Exemple 1

```
@use "sass:map";

$font-size: (logo:7rem, heading:5rem,
project-heading:4rem, label:2rem);

//Retourne une valeur de la map $font-size
en fonction de sa clé
@function getFontSize($key) {
    @return map-get($font-size, $key);
}

@debug getFontSize(heading); //5rem
```

Exemple 2

```
//Retourne la largeur d'un bloc en fonction
de la largeur et du nombre de colonnes et de
la largeur de la gouttière

$largeur-colonne: 60px;
$largeur-gouttiere: 20px;
@function largeur-grille($nb-colonne) {
    @return $nb-colonne * $largeur-colonne +
($nb-colonne - 1) * $largeur-gouttiere;
}

body {
  width: 940px;
}
```

DIFFERENCE ENTRE LES FONCTIONS ET LES MIXINS

La différence entre une mixin et une fonction est le résultat retourné une fois compilé.

Une mixin peut générer plusieurs lignes de code CSS tandis qu'une fonction vous retournera uniquement une valeur.

Grâce aux fonctions, il vous sera possible de calculer le pourcentage d'un bloc, de convertir une unité « px » en « em/rem » ou encore d'éclaircir une couleur.

On peut comparer cela à des procédures et des fonctions.

ATELIER

Créer une fonction simplifiée de calc()

FACTORISER LE CODE

DECOUPER SON CODE DANS PLUSIEURS FICHIERS

Permet de séparer votre fichiers css en plusieurs petits fichiers.

Cela permet de modulariser le CSS et d'en faciliter la maintenance.

Préfixer le nom de fichier avec un _ .

Les fichiers préfixés d'un underscore sont appelés des partials.

Exemples :

- _reset.scss
- _variables.scss
- _header.scss

IMPORTER DES PARTIALS

Importer avec la directive @import nom de fichier sans l'extension .scss

Exemple : @import '_reset';

Les partials ne créent pas de nouveau fichier lors de la compilation, elles sont intégrées dans le fichier où a lieu l'import.

IMPORTER DES FICHIERS CSS

Outre l'importation de fichiers .sass et .scss, Sass peut importer de simples fichiers .css.

La seule règle est que l'importation ne doit pas inclure explicitement l'extension .css, car celle-ci est utilisée pour indiquer un @import CSS.

```
//Avec extension .css
@import 'normalize.css’;

//Résultat à la compilation
@import 'normalize.css';
```

```
//Sans extension
@import 'normalize’;

//Résultat à la compilation
Le code est compilé
```

LES PLACEHOLDERS

Sass dispose d'un type de sélecteur particulier, appelé " placeholder ". Il ressemble beaucoup à un sélecteur de classe et se comporte comme tel, mais il commence par un % et n'est pas inclus dans la sortie CSS.

SCSS

```scss
%myheading{
    font-weight: bold;
    font-size: 16px;
  }

  h1 {
    @extend %myheading;
    font-size: 20px;
  }
  h2 {
    @extend %myheading;
    font-size: 18px;
  }
```

CSS

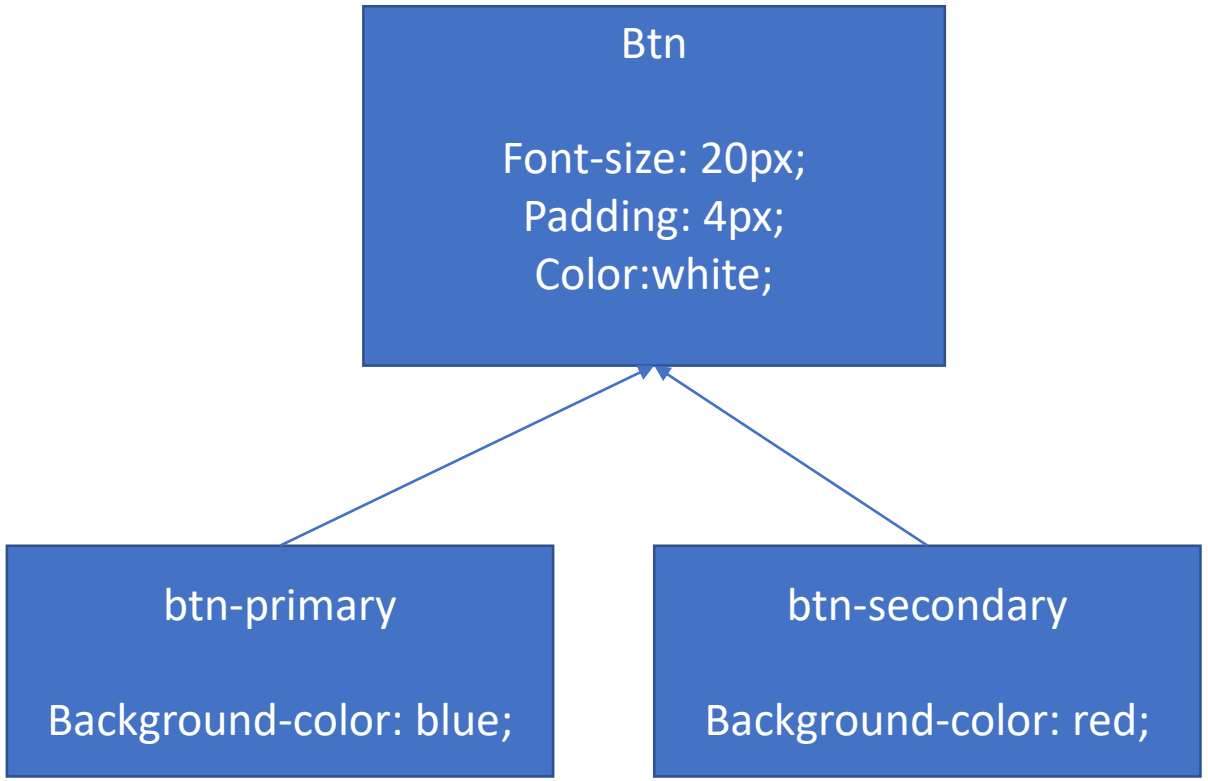
```css
h2, h1 {
  font-weight: bold;
  font-size: 16px;
}

h1 {
  font-size: 20px;
}

h2 {
  font-size: 18px;
}
```

ETENDRE LES SELECTEURS @EXTEND

L’ héritage permet d’étendre les sélecteurs ou classes.



L’héritage permet d’hériter des propriétés et valeurs de son parent.

L’héritage se fait grâce à la directive @extend

ETENDRE LES SELECTEURS @EXTEND

Exemple d'héritage d'un bouton

SCSS

```scss
@use "sass:color";
$blue-color: blue;
$red-color: red;

.btn{
    font-size: 30px;
    padding: 10px;
    color: white;
}

.btn-primary{
    @extend .btn;
    background-color: $blue-color;
    &:hover{
        background-color: darken($blue-color, 20%)
    }
}

.btn-warning{
    @extend .btn;
    background-color: $red-color;
    &:hover{
        background-color: darken($red-color, 20%)
    }
}
```

CSS

```css
.btn, .btn-warning, .btn-primary {
  font-size: 30px;
  padding: 10px;
  color: white;
}

.btn-primary {
  background-color: blue;
}
.btn-primary:hover {
  background-color: #000099;
}

.btn-warning {
  background-color: red;
}
.btn-warning:hover {
  background-color: #990000;
}
```

DIFFERENCE ENTRE @EXTEND ET @MIXIN

@mixin duplique un ensemble de règles alors que @extend duplique le sélecteur, la classe ou l'identifiant.

CSS provenant d'une mixin

```
.btn-primary {
  font-size: 20px;
  padding: 4px;
  color: white;
  background-color: blue;
}

.btn-warning {
  font-size: 20px;
  padding: 4px;
  color: white;
  background-color: red;
}
```

CSS provenant d'un extend

```
.btn, btn-primary, .btn-warning {
  font-size: 20px;
  padding: 4px;
  color: white;
}

.btn-primary {
  background-color: blue;
}

btn-warning {
  background-color: red;
}
```

ATELIER

Répartir notre code dans plusieurs fichiers dédiés

CRÉER DES MODULES

LES MODULES SAAS

Sass fournit de nombreux modules intégrés qui contiennent des fonctions utiles (et un mixin occasionnel).

Toutes les URL des modules intégrés commencent par sass : pour indiquer qu'ils font partie de Sass lui-même.

Quelques modules Sass :

- ✓ sass:math fournit des fonctions qui opèrent sur les nombres.
- ✓ sass:string permet de combiner, de rechercher ou de séparer facilement des chaînes de caractères.
- ✓ sass:color génère de nouvelles couleurs à partir des couleurs existantes, ce qui facilite la création de thèmes de couleurs.

ESPACE DE NOM ET ALIAS

Choix d'un espace de nom

Par défaut, l'espace de noms d'un module est simplement le dernier composant de son URL sans extension de fichier.

Cependant, il peut arriver que vous souhaitiez choisir un espace de noms différent : vous pouvez utiliser un nom plus court pour un module.
Vous pouvez le faire en écrivant @use "<url>" as <namespace>.

Exemple

@use custom as c;

PRESENTATION DE LA REGLE @USE

Les feuilles de style chargées par @use sont appelées "modules".

La directive @use est similaire à @import, mais présente quelques différences notables :

- ✓Le fichier n'est importé qu'une seule fois, quel que soit le nombre de fois où vous l'utilisez dans un projet.
- ✓Les variables, les mixins et les fonctions (ce que Sass appelle les "membres") qui commencent par un trait de soulignement (_) ou un trait d'union (-) sont considérés comme privés et ne sont pas importés.
- ✓Les membres du fichier utilisé (buttons.scss dans ce cas) sont uniquement disponibles localement, mais ne sont pas transmis aux importations futures.
- ✓Tous les membres importés sont nommés par défaut.

DEFINIR DES VALEURS PAR DÉFAUT

Variable par défaut, mettre le drapeau !default.

Exemple :

- $medium : 768px!default;
- $large: 1024px!default;

Cela permet d’indiquer une valeur si elle n’est pas définie ou si la valeur est null.

CHARGER UN MODULE

Pour charger un module, utiliser la directive @use.

Les règles @use d'une feuille de style doivent venir avant toutes les règles. Toutefois, vous pouvez déclarer des variables avant les règles @use pour les utiliser lors de la configuration des modules.

```scss
@use "sass:color";

.button {
    $primary-color: #6b717f;
    color: $primary-color;
    border: 1px solid color.scale($primary-color, $lightness: 20%);
}
```

```css
.button {
  color: #6b717f;
  border: 1px solid #878d9a;
}
```

RENDRE DES PROPRIETES PRIVEES

Sass vous permet de définir facilement un membre privé en commençant son nom par - ou _. Ces membres fonctionneront normalement dans la feuille de style qui les définit, mais ils ne feront pas partie de l'API publique d'un module. Cela signifie que les feuilles de style qui chargent votre module ne peuvent pas les voir !

```scss
// src/_corners.scss
$-radius: 3px;

@mixin rounded {
  border-radius: $-radius;
}
```

```scss
// style.scss
@use "src/corners";

.button {
  @include corners.rounded;

  // Erreur, $-radius n’est pas visible en dehors de _corners.scss
  padding: 5px + corners.$-radius;
}
```

CONFIGURER LES VARIABLES A L'IMPORT

Une feuille de style peut définir des variables avec le drapeau !default pour les rendre configurables. Pour charger un module avec configuration, écrivez @use <url> avec (<variable> : <value>, <variable> : <value>). Les valeurs configurées remplaceront les valeurs par défaut des variables.

```scss
// _library.scss
$black: #000 !default;
$border-radius: 0.25rem !default;
$box-shadow: 0 0.5rem 1rem rgba($black, 0.15) !default;

code {
    border-radius: $border-radius;
    box-shadow: $box-shadow;
}
```

```scss
// style.scss
@use "library" with (
    $black: #222,
    $border-radius: 0.1rem
);
```

```scss
//css
code {
    border-radius: 0.1rem;
    box-shadow: 0 0.5rem 1rem rgba(34, 34, 34, 0.15);
}
```

ATELIER

Créer, importer et paramétrer un module

ARCHITECTURE

Patron 7-1

Le pattern 7-1 permet une meilleure organisation de votre code.

7 dossiers - 1 fichier

- ✓base/
- ✓components/
- ✓layout/
- ✓pages/
- ✓themes/
- ✓abstracts/
- ✓vendors/
- ✓main.scss

Composition des fichiers

- ✓**base**/ style des éléments de base, ex: reset, typography
- ✓**layout**/ système de grille, disposition. ex:navigation, grid, header, footer, sidebar, forms
- ✓**components**/ vision plus locale que le layout, penser «widget». ex: button, carousel, cover, dropdown
- ✓**pages**/ en cas de style spécifique à certaines pages. ex: home, contact
- ✓**themes**/ ex: theme, admin
- ✓**abstracts ou utils**/ex: variables, functions, mixins, placeholders
- ✓**vendors**/ style externe. ex: bootstrap, jquery-ui

ATELIER

Organiser les fichiers selon le pattern 7-1

MÉTHODOLOGIE BEM

Pour avoir un code CSS plus facilement maintenable, organisé et structuré, vous pouvez utiliser des conventions de nommage comme BEM

Block

Element

Modifier

- .block représente un bloc
  - .header
- .block__element représente un élément du bloc
  - .header__titre
- .block__element--modifier représente une modification de l'élément
  - .header__titre--blue

Documentation : (https://en.bem.info/)

AUTRES MÉTHODOLOGIE DE NOMMAGE

Object-Oriented CSS (OOCSS) est une autre méthodologie de nommage des feuilles de styles

Le principe d'OOCSS est de séparer la structure et l'apparence.
Le fait de séparer les deux permet de créer des éléments réutilisables (des objets).

On repère les objets/patterns visuels pour en faire des classes réutilisables afin de rendre le CSS indépendant de la structure HTML.

Documentation : (http://oocss.org/)

Il existe d'autres méthodologies comme SMACSS (http://smacss.com/), par exemple